Tutorial 03 - Amortized, expected, and average-case analyses; sorting

CS 240E Winter 2023

University of Waterloo

Monday, January 30, 2023

1. **Amortized analysis.**
   We are given a binary search tree on n nodes, storing n distinct keys. We can list all keys in increasing order using in-order traversal in time linear in n.

   The operation *successor*(x) returns the in-order *successor* of x in the tree, which is the node z with $x.key < z.key$ and no other keys are stored in between (or `null` if such z does not exist), in Θ(height of the tree) time.

   Consider the algorithm to print all keys in the tree T in increasing order:

   ```
   x = T.get_min()
   print(x.key)
   while(T.successor(x) is not null):
       x = T.successor(x)
       print(x.key)
   ```

   (a) Give an asymptotic bound on the worst-case runtime of this algorithm, if the height of T is in $\Theta(\log n)$.

   (b) Show that the amortized runtime of *successor* is $O(1)$ (and therefore the runtime of the algorithm is $\Theta(n)$).

2. **String comparison.**
   Let A and B be two bitstrings of length n (modelled here as arrays where each entry is 0 or 1). A *string-compare* tests whether A is smaller, larger, or the same as B and works as follows:

   Show that the average-case run-time of *str-cmp* is in $O(1)$. You may use without proof that $\sum_{i \geq 0} \frac{i}{2^i} \in O(1)$.

Algorithm 1: *str-cmp*(A, B, n)

```
for (i = 0; i < n; i++) do
    if A[i] < B[i] then return "A is smaller"
    if A[i] > B[i] then return "A is bigger"
return "They are equal"
```

3. **Average-case vs expected (hiring problem).**
   Suppose we must hire a new employee. There are n candidates arriving sequentially, one each day.

   It takes I time units to interview a candidate, and it takes H units to hire them.

   We want to have at all times the best possible person for the job. After interviewing each applicant, if they are better than our current employee, we hire them immediately (and fire our current employee).

   We can compare two candidates in constant time.

```
hire(cand[1..n]):
    curr = dummy candidate  // compares worse than anyone
    for i = 1..n:
        interview cand[i]
        if cand[i] is better than curr:
            hire cand[i]
            curr = cand[i]
```

   Suppose m candidates are hired. Then the worst-case runtime is in $\Theta(In + Hm)$.

   We can rank each candidate with a unique number between 1 and n and use $rank[i]$ to denote the rank of candidate i. We adopt the convention that a higher ranked applicant corresponds to a better qualified applicant.

   Note that the ordered list

$$\langle rank[1], \ldots, rank[n] \rangle$$

   is a permutation of the list $\langle 1, \ldots, n \rangle$.

(a) Describe an instance that achieves the runtime $\Omega(Hn)$.

(b) Show that in the **average-case** we hire a new candidate $O(\log n)$ times.

4. **Morris's probabilistic counting.**
With a deterministic b-bit counter, we can only count up to $2^b - 1$. With *probabilistic counting* we can count to *larger values* at the expense of *loss of precision.*

We let a counter reading of i represent a count of v_i, for $0 \leq i \leq 2^b - 1$. Initially the counter reads 0, indicating the count of $v_0 = 0$.

The operation *increment* works on a counter with reading i in a probabilistic manner:

- if $i < 2^b - 1$, increase counter reading with probability

$$\frac{1}{v_{i+1} - v_i},$$

and leave the counter unchanged otherwise.

- if $i = 2^b - 1$, report overflow.

Note that if we select $v_i = 1$, then the counter is an ordinary deterministic counter. More interesting situations arise if $v_i = 100i$, $v_i = 2^i$, or $v_i = i$-th Fibonacci number.

Assume that the probability of an overflow is negligible. Show that the value represented by the counter after n *increment* operations is n.

5. **Partially Sorted.**
Let $0 < \epsilon < 1$. Suppose that we have an array A of n items such that the first $n - n^\epsilon$ items are sorted. Describe an $O(n)$ time algorithm to sort A.